原作者： X-Plane Team

翻译： PeterShen （Skysim Team）

译者团队网站： www.skysim.cc

日期 ：2017.5.10

翻译版本 ：1.1

如果你已经在版本10中使用过并想了解版本11中的改动，请看此本快速迁移手册。

X-Plane 11的新消息：

用于加载飞行器的ACFN命令，定位飞行器的PREL命令，加载定位飞行器的ACPR命令已配置了新的结构体，具体见文档说明。

X-Plane 11停用的消息：

PAPT	place by airport.	改用 PREL.
PMAP	place by airport.	改用 PREL.
VEH1	移动单架飞机.	改用 VEHX.
VEHA	移动所有飞机.	改用 VEHX.
MENU	选择一个菜单.	改用 CMND.
CHAR	选择一个菜单.	改用 CMND.
MOUS	选择一个菜单.	改用 CMND.
ISET	打开或关闭一个英特网输出.	改用 ISE4.
ACFN	旧的ACF结构体没有livery.	改用 ACFN，livery可选项见文档.
HACF	炸掉一个飞行器.	不再支持.
R_QT	播放一个quicktime电影.	不再支持.

X-Plane的驱动与监听

此文档将讲述如何获取、发送数据到X-Plane 11。你允许任意处理使用到的数据 ,包括用于模拟舱，动平台的制造等，但通讯需要使用作者规范的消息去与X-Plane交互，如何交互不受作者约束。

与X-Plane11的UDP交互可以通过以太网或WIFI。另外，亦可使用X-Plane插件进行交互，插件可以更快的控制更多参数，不过编写插件需要一定的学习成本。作者推荐使用插件去读写大量的数据，这样可以得到更快的响应速度。你也可以不使用插件，使用UDP简单的读写一些数据，以建立你自己的程序。

本文需要你对UDP消息机制有一定的了解，对具体UDP消息的描述才是本文的重点。

提示

提示：X-Plane总是从编号49000的接口接收数据。

提示：所有字符串需要以空值（NULL）结尾。

提示：打开 【Setting】菜单，找到 【General】 标签， 打开 【Output network data to Log.txt】勾选框，可以采集通讯的消息记录，打开输出会拖慢一些模拟器速度，不过作为调试用可以打开一会以便观察数据。

译者提示：严格按照文档所标注的格式传递消息，**尤其是成员的顺序。**

开始使用

使用X-Plane数据交互有两个用途：

1、将X-Plane作为自己仿真模型的视景使用。

2、将X-Plane仿真的数据输入到自己的系统当中。

输出数据驱动外部程序: RPOS

用途：从X-Plane获取到飞机现在的经纬度、海拔、航向、姿态参数。

接口：主机地址，UDP 49000。

RPOS（4char + NULL，共五字符）加上一个字符串（同样需要以NULL结尾）。字符串包含一个数字，说明你希望X-Plane每秒给你回馈多少次信息。

发送 RPOS_60_（**_代表了NULL**），说明X-Plane每秒将发送60次RPOS消息。

注意：发送该指令后，X-Plane会立刻开始反馈，反馈地址为发送请求的IP和端口。

四字符 RPOS 与一个 NULL	
double dat_lon	经度，度
double dat_lat	纬度，度
double dat_ele	海拔高度，米
float y_agl_mtr	对地高度，米
float veh_the_loc	pitch，度
float veh_psi_loc	真实航向，度
float veh_phi_loc	roll，度
float vx_wrl	X轴速度，向东，米/秒
float vy_wrl	Y轴速度，向上，米/秒
float vz_wrl	Z轴速度，向南，米/秒
float Prad	roll率，弧度/秒
float Qrad	pitch率，弧度/秒
float Rrad	yaw率，弧度/秒

获取天气雷达信息: RADR

用途：输出数据可以让你建立自己的气象雷达。

接口：主机地址，UDP 49000。

五个字符RADR（4char + NULL，共五字符）加上一个字符串（同样以NULL结尾），声明了X-Plane每秒向你发送多少个雷达数据点。

发送 RADR_10_（**_代表了NULL**），说明X-Plane每秒向你发送10个数据点。

注意：发送该指令后，X-Plane会立刻开始反馈，反馈地址为发送请求的IP和端口。

四个字符 RADR 和一个 NULL	
float lon	雷达点经度
float lat	雷达点纬度
float storm_level_0_100	降水量，0到100
float storm_height_meters	风暴平面的海拔高度，米

利用现有数据驱动视景: VEHX

用途：利用X-Plane提供更好的视景画面，运行你仿真出的模拟数据。

接口：主机地址，UDP 49000。

五个字符 VEHX（四个字符加一个NULL）加上数据，数据格式已格式化过，数据间没有空格。

int p	飞机序号，0为主飞机
double dat_lat	纬度，度
double dat_lon	经度，度
double dat_ele	海拔高度，米
float veh_psi_true	航向，度
float veh_the	pitch，度
float veh_phi	roll，度

其他消息

下面是X-Plane所定义的数据类型：

XCHR	字符
XINT	4 byte int
XFLT	4 byte int 和 float
XDOB	双精度浮点数

（以上数据类型转化根据机器平台所定）

有时候直接使用字符串去替代数字直接传递，避免了byte转化的问题。

所有的UDP包都采用相同的格式：

5字符的开头（定义消息的类型），然后加上传递数据（包含了你想要发送或者接收的数据）。开头的五个字符中，前四个是char，第五个是一个空值，后面跟上需要发送的数据。

所以，想要给X-Plane发送UDP消息，就发送：

-四个字符的标签

-一个byte的空值

-你想要发送的消息的数据

注意结构体：

你可能通过UDP发送或接收多个结构体，所以记得在代码中声明。

包括在Mac上使用时，结构体字节必须与Intel 64-bit X-Code对齐。对齐方式是：四字节为int或float，八字节为double。所以，一个int也可能占用八个字节，如果结构体中存在一个八字节int，那下一个变量是一个double！

切记，如果结构体中含有double类型的参数，那么一个4byte int的数组中可能包含8byte的double。每个变量还是占用4byte，交互时还是以4byte为分界，只有最初的参数是以8byte来处理的。

不同开发者使用不同编辑器与编辑选项，经常使得数据格式填充不完整，最后导致数据长度错误。作者建议刚开始调试时，打开设置中的调试窗口查看交互的数据有无问题。

允许发送的数据

以下的数据结构体是允许被发送的（同一为5字符的描述开头）。

加载一架飞机：ACFN

```
struct acfn_struct
{
        xint            acfn_p                  ;
        xchr            acfn_path_rel[150]      ;
        xint            livery_index            ;
};
```

使用这个结构体去加载飞机。

acfn_p是加载的飞机，path是飞机的路径，发送以上数据可以加载预定的飞机。发送1到19可以加载其他飞机。

指定地点初始化飞机：PREL

```
struct PREL_struct
{
        init_flt_enum type_start  ;// 见下表说明
        xint          p_idx       ;// 飞机序号，0为自己，1-19为其他飞机
        xchr          apt_id[idDIM] ;// 机场ID，需要查询机场的ID
        xint          apt_rwy_idx ;// 跑道序号，需要查询ID
        xint          apt_rwy_dir ;// 跑道方向，需要跑道序号支持
        xdob          dob_lat_deg ;// 经纬高度指定Start，纬度
        xdob          dob_lon_deg ;// 经度
        xdob          dob_ele_mtr ;// 高度
        xdob          dob_psi_tru ;// 航向，度
        xdob          dob_spd_msc ;// 速度，米/秒
};
```

使用此参数可以让飞机置位在某个地点。

参数Start的值如下：

```
loc_repeat_last     5     // 给ATC或重置飞行使用
loc_specify_lle     6     // 给地图使用
loc_general_area    7     // 自动加载飞机并增加飞机数量
loc_nearest_apt     8     // 加载新飞机，不改变位置
loc_snap_load       9     // 从快照中加载，不改变航路与飞机位置

loc_ram             10    // 斜坡开始
loc_tak             11    // 跑道上起飞
loc_vfr             12    // VFR进近
loc_ifr             13    // IFR进近

loc_grs             14    // 草地跑道
loc_drt             15    // 脏跑道
loc_grv             16    // 砂石跑道
loc_wat             17    // 水上机位
loc_pad             18    // 直升机停机坪
loc_cat             19    // 舰载平台
```

loc_tow	20	// 滑翔机，牵引机
loc_win	21	// 滑翔机，卷扬机
loc_frm	22	// 编队飞行
loc_Are	23	// 重新加油- Boom
loc_Nre	24	// 重新加油 - Basket
loc_drp	25	// B52投掷
loc_pig	26	// 坨着航天飞机
loc_car	27	// 运输机进近
loc_fri	28	// 护卫舰进近
loc_rig	29	// 少油进近
loc_pla	30	// 重油进近
loc_fir	31	// 森林火灾进近
loc_SO1	32	// 航天飞机
loc_SO2	33	// ""
loc_SO3	34	// ""
loc_SO4	35	// ""
loc_shuttle_glide	36	// 只能用于抛投坨着的航天飞机

通常使用 loc_specify_lle 去简单指定经度，纬度，高度加载新飞机。
此情况下你可以将机场参数设置为0。

加载飞机并指定位置： ACPR

发送ACPR加上一个NULL，跟上两个上述的结构体即可。

运行一个指令: CMND

数据输入结构体是一个字符串

初始化消息的数据非常简单！
打开设置，可查看X-Plane的指令，指令被组别化命名了。例如，指令格式为none/none。

典型的获取襟翼抬升的消息如下

CMND0+sim/flight_controls/flaps_up

发回一个我希望获取到的值: RREF

发送5字符的指令 RREF (NULL结尾) 加上一个结构体:

```
struct dref_struct_in
{
        xint dref_freq          ;
        xint dref_en                    ;
        xchr dref_string[400]   ;
};
```

dref_freq说明每秒发送几次数据。

dref_en是一个integer参数，为了说明反馈的是哪个dataref，通常你会请求很多个参数。

dref_string说明你需要什么dataref。

如果dataref是一个数组（比如引擎推力，可能有8个引擎），就在后面加一个[xxx]，xxx是序号。[与]中应简单的包含一个数字去指定你需要的序号。

所以，发送 sim/flightmodel/engine/POINT_thrust[1]，可以得到第二号引擎的数据（起始编号为0）。X-Plane将立刻反馈到你发送指令的IP端口。

你将得到：

```
struct dref_struct_out
{
        xint dref_en    ;
        xflt dref_flt   ;
};
```

dref_en说明了dataref在结构体中处在的序号。

dref_flt是dataref值，有可能是浮点数，也可能是整数。

这样你可以获取任何你想要的参数。

发动一个为零的dref_freq的参数停止收取dataref反馈。

给一个Dataref发送数据： DREF

```
struct dref_struct
{
        xflt var;
        xchr dref_path[500];
};
```

使用这个参数设置任意的data-ref。你可以发送任意的float值给整个模拟器中任意的data-ref。可以前往http://www.xsquawkbox.net/查看有关dataref的东西。

注意：一个NULL字符必须放置在dref_path的结尾，否则会失败，你发送的消息应该是以下格式：
DREF0+（4byte的var值）+dref_path+0+多余的NULL填满整个509byte的消息
（0应为NULL）

以下是打开除冰开关的样例：

DREF0+(4byte的值，为1)+ sim/cockpit/switches/anti_ice_surf_heat_left+0+多余的NULL填满整个509byte的消息

样例请整体作为一个数据发送。
记住：你可以打开Setting菜单，打开diganostics选项以便查看输出的数据与X-Plane想获取的数据。

将数据输出设置为值： DATA

在输出数据的同时你还可以设置输出数据的值(不是所有的值都可以改变 ,比如马赫数就不可以改动)。

在从UDP输入参数或控制指令时，X-Plane会使用它们作为模拟数据操作，你可以使用自己的硬件去发送手柄的控制指令从而操控飞机。当立即回传的值改变时，有可能是X-Plane复写了这个参数。

```
struct data_struct
{
        int index;          // 数据编号
        float data[8];      // 最高8byte的数据输出，很多值都不会占用整个8byte
};
```

发送一个-999意味着你不利用这个值，或将操作恢复默认。

所以想发送一个数据消息去控制一些值，要发送：
DATA（4字节）
0（0为一个字符）
data_struct（数据结构体部分，4byte对齐）

使用以上方式可以对X-Plane中的一些值进行控制。

选择参数显示或不显示在座舱界面：DSEL/USEL/DCOC/UCOC

数据输入结构体是XINT的一个序列。

例如你在驱动一个运动平台时，将有很多的调用的参数显示在屏幕上，这时你只想有个把参数输出到UDP上就可以使用DSEL，属性为数据的序号，1是列表中第一个，2是第二个，以此类推。

所以，DSEL0456请求了第四个、第五个和第六个数据，它们每秒发送到你的IP端口上。DSEL本身是字符串，但是4、5、6需参考你机器的编码方式。

使用DSEL选中输出的数据。
使用USEL取消选中输出的数据。
使用DCOC将数据显示到屏幕而不发送到UDP。
使用UCOC取消显示屏幕数据也不发送到UDP。

设置英特网选项：ISE4

此参数允许你直接设置X-Plane的英特网选项。当你有多个分机显示并不想手动去设置每一台机器，这个选项就变得十分有用处。

结构体如下：

```
struct ISE4_struct                                  // 一个IPv4设置
{
        xint index                  ;               // 发送方式的枚举
        xchr snd_ip_str[16]         ;               // 目的IP地址，英文书写
        xchr snd_pt_str[ 8]         ;               // 目的端口，字符串
        xint snd_use_ip             ;               // 使用多个IP
};
```

下面是X-Plane 11.00的发送方式枚举：

```
    if(input<=18)sel=ip_mplayer_00          +input      ;       // 多人
else if(input<=38)sel=ip_exvis_00           +input-19   ;       // 额外的图形化界面
else if(input==39)sel=ip_master_is_exvis                ;       // 主机是个额外的图形化界面
else if(input==42)sel=ip_master_is_IOS                  ;       // 主机是个IOS设备
else if(input==62)sel=ip_IOS_is_master                  ;       // IOS是主机
else if(input==64)sel=ip_DOUT_ui_set                    ;       // 数据输出目标
else if(input==71)sel=ip_Xavi_1                         ;       // Xavion 1
else if(input==72)sel=ip_Xavi_2                         ;       // Xavion 2
else if(input==73)sel=ip_Xavi_3                         ;       // Xavion 3
else if(input==74)sel=ip_Xavi_4                         ;       // Xavion 4
else if(input==75)sel=ip_fore_ip_addy                   ;       // Foreflight，单播
else if(input==76)sel=ip_fore_broadcast                 ;       // Foreflight，广播
else if(input==77)sel=ip_control_pad                    ;       // 给IOS提供的X-Plane操纵界面
```

设置英特网选项：ISE6

此参数允许你直接设置X-Plane的英特网选项。当你有多个分机显示并不想手动去设置每一台机器，这个选项就变得十分有用处。

结构体如下：

```
struct ISE6_struct                          // 一个IPv6设置
{
        xint index                  ;
        xchr snd_ip_str[46]         ;       //目的IP地址，英文书写
        xchr snd_pt_str[ 6]         ;       // 目的端口，字符串
        xint snd_use_ip             ;       // 使用多个IP
};
```

与IPv4使用的类型枚举一致。

播放一个音频：SOUN

```
struct soun_struct          // play any sound
{
        xflt freq,vol               ;
        xchr path[500]              ;
};
```

播放一段WAV音频，在结构体中输入WAV文件的路径。频率和音量大小范围在0.0到1.0之间。

播放一个循环音频：LSND 和 SSND

```
struct loop_struct
{
        xint index                          ;
        xflt freq,vol               ;
        xchr soun_path[500]         ;
};
```

循环播放一段WAV音频，序号从0到4（总共可以播放五段音频）。

LSND开始播放，SSND停止播放。

加载一个对象：OBJN

```
struct objN_struct // object name: draw any object in the world in the sim
{
        xint index              ;
        xchr path[500]          ;
};
```

就像飞机的结构体一样，不过内容为OBJ7对象（可以查看San Bernadine的“KSBD_example.obj”作为OBJ7的例子）。
使用这条消息可以让X-Plane显示任何你想要显示的对象。显示的位置使用下面这条消息。

放置一个对象：OBJL

```
struct objL_struct                      // 对象位置
{
        xint index              ;
        xdob lat_lon_ele[3]     ;
        xflt psi_the_phi[3]     ;
        xint on_ground          ;       // 若想将对象至于地面上，将值设置为0
        xflt smoke_size         ;       // 若对象在冒烟，将值设置为冒烟的大小
};
```

你可以放置一辆坦克，一颗发射的导弹与任何你想得到的东西。

在X-Plane中发送一个提示消息：ALRT

```
struct ALRT_struct                                      // 发送一个提示消息
```

```
{
public:
        ALRT_struct(){memset(this,0,sizeof(*this));}
    ~ALRT_struct(){}

        xchr m_m1[240];                                    // 需要8byte对齐
        xchr m_m2[240];                                    // 需要足够长度去承载消息
        xchr m_m3[240];
        xchr m_m4[240];
};
```

故障系统：FAIL

故障系统，数据指定了哪个系统产生故障。发送ASCI字符串（例如“145”）选择系统，0是第一个序号。故障列表请查阅X-Plane（现在为vacuum system），并以1作为单位累加。

恢复系统：RECO

恢复系统，数据指定了恢复哪个系统故障。发送ASCI字符串（例如“145”）选择系统，0是第一个序号。故障列表请查阅X-Plane（现在为vacuum system），并以1作为单位累加。

导航设备失效：NFAL

失效一个导航设备，内容为导航设备的ID序号。

恢复导航设备：NREC

恢复一个导航设备，内容为导航设备的ID序号。

恢复所有的系统故障：RESE

发送RESE0（0为NULL）恢复所有已产生的系统故障。

退出指令：QUIT 和 SHUT

QUIT (不需要任何消息跟在此指令之后)
SHUT (不需要任何消息跟在此指令之后)

使用Beacon探索X-Plane

为了发送与接收X-Plane的UDP消息，你必须知道网络内设备的IP地址。你可以手动指定这些设备的地址，或者你也可以使用Beacon消息，让网络中运行X-Plane的设备发送一个广播宣告。

Beacon消息使用了多播技术 ,这项技术使得你可以收到特定网段中运行X-Plane的设备发送一个宣告。Beacon可以使用在同一机器或本地局域网中。

为了使用Beacon消息 ,你需要加入多播组239.255.1.1并监听端口49707。这看起来像一个IP地址，但却不在网络中，这是一个组播地址标记。根据不同的机器，查看关于如何多播UDP消息，例如setsockopt()功能和IP_ADD_MEMBERSHIIP参数。

当你设置了一个socket去接收X-Plane的多播消息，你可能使用SO_REUSEADDR（Mac中为SO_REUSEPORT）选项，使得同一机器上不同的程序都可以收到Beacon消息。如果你没有使用SO_REUSEADDR（Mac中为SO_REUSEPORT），每台机器只有一个程序会被X-Plane检测到，其他程序会忽视socket消息。一般来说不会使用这些参数，除非你非常了解所做的东西。切记不要将SO_REUSEADDR或SO_REUSEPORT设置给端口49707接收Beacon消息。

当你收到了一个Beacon消息，结构体应该如下所示：

5个字节的消息开头，BECN\0

```
struct becn_struct
{
        uchar beacon_major_version;    // 一次发送一个，主版本号
        uchar beacon_minor_version;    // 一次发送一个，副版本号
        xint application_host_id;      // 1是X-Plane，2是PlaneMaker
        xint version_number;                    // 104103是X-Plane 10.41r3
        uint role;                              // 1为主机，2是额外显示，3是IOS
        ushort port;                            // X-Plane监听的端口，49000为默认值
        xchr    computer_name[500];             // 电脑的主机名，例如Joe's Macbook
};
```

解析这个结构体可以让你找到网络中运行的各个X-Plane，查看机器的主机名，是否为主机或是视角从机，最后可以看到端口是否为默认的49000。

如果你能够读取结构体说明对方主版本与你是一致的，如果失败则有可能你收到的beacon_major_version与你是不一致的。